Aligning Web System and Organisational Models

Andrew J. Bucknell, David Lowe, Didar Zowghi
University of Technology, Sydney
P.O. Box 123, Broadway NSW 2007, Australia
{andrew.j.bucknell, david.lowe, didar.zowghi@uts.edu.au}

Abstract

A significant area of research is the alignment of web system and organisational requirements. In this paper we describe an approach to facilitating this alignment using graphical models of the processes that are being supported by a web-based system. This approach is supported by the AWeSOMe modelling architecture. This architecture allows us to investigate the effectiveness of different notations for modelling systems. The architecture is being implemented as the AWeSOMe modelling tool, which will be used to investigate our approach to alignment in industry-based case studies.

Introduction

Current software development is often characterised by significant early uncertainty in system scope, particularly where the Web is leveraged in supporting changes to business processes. This uncertainty can then lead to poor alignment between business processes and IT systems, substantial ongoing system redevelopment, and client and customer dissatisfaction with the resultant systems. The lack of appropriate approaches to address these problems leads to significant costs in web development projects [11], which is a significant business activity in its own right [6], with cost overruns being a significant waste of resources.

The AWeSOMe tool described in this paper is being developed to support research which investigates an innovative approach to reducing this uncertainty and the resulting volatility, and through this support a more rapid resolution of the development scope for software systems. Specifically, the research will utilise recent progress in the development of high-level modelling approaches [14, 16] (which more effectively link system information management and functional behaviours with business processes) to enable the automated identification of potential discordances between the software system being developed and the organisational context in which the system exists. These discordances arise when aspects of the (proposed) system or business processes are changed without appropriate consideration of the impacts on the complex inter-relationships which exist within the composite software/business environment.

Specific objectives of the research include:

Customisation and extension of existing modelling approaches to support identification of key discordances: Existing modelling approaches can represent the relationship between software systems and business domains [15], but have not traditionally been used to identify or reason about discordances in these relationships when changes are made to either of these (see the section below for an explanation of this issue). We aim to adapt these modelling approaches to allow this reasoning about potential discordances to occur. The AWeSOMe tool will allow us to prototype extensions to modelling notations and use these notations to model systems.

Development of algorithms for automated identification of discordances: Once the modelling approaches have been developed, we will develop algorithms to allow reasoning about the models and subsequent automated identification and appropriate reporting of discordances. The AWeSOMe tool will support the integration of software components implementing algorithms that analyse the underlying data model for discordances.

Evaluation of the effectiveness of the techniques: The AWeSOMe tool can be used in case studies that demonstrate the modelling and reasoning algorithms and allows evaluation of the resultant impact on the reduction of scope volatility during the development process. Particular attention will be payed to applicability of the approaches across a range of problem domains and technologies, and the applicability to managing real-world projects.

This paper describes the AWeSOMe tool which is being developed to support research into aligning web system and organisation models. We begin by discussing the existing research that is informing this work. We then discuss the problem that this research seeks to address. Next we discuss the research methodology that is being applied to investigate this problem, and discuss the role of the AWeSOMe tool in this research. Next we

discuss the conceptual and software frameworks that are supporting the development of AWeSOMe. We conclude by outlining the next stages of our work building on the AWeSOMe tool.

Background

This project will build upon our earlier research in several key areas. The first area, related to the earlier stages of the Web development process, includes research into Web characterisation models [11], development processes [10], requirements volatility[22], Web Impact analysis [12, 20, 21] and – most recently – our work on the role of issue resolution in supporting the determination of system scope in Web projects. This last body of research is crucial in the context of this project. For the commercial Web projects which we studied, once an initial project brief had been established the key trigger for almost all subsequent adjustments to the project scope was the resolution of issues. These issues related to the way in which the system environment impacts on, and is impacted (or changed) by the introduction of the system. This leads us to an approach to scope refinement which will be based on the dualistic modelling of the system and environment as they currently exist prior to the development (an *as-is* view), and as they are desired to be (a *to-be* view).

The second area of our previous research which feeds into this project is our development of high-level information flow models [16]. Whilst the detailed behaviour and low-level information design of a Web system tends to be extremely volatile during (and usually after) the system development the flow of information between the system and its environment tends to be much more stable. We subsequently showed [17] that a representation based on high-level information flows provided a clearer basis for system design, and – most significantly – facilitated identification of ways in which a proposed design change may result in impacts upon the systems environment. Further, the information flow models, when combined with conventional work flows (a simplified form of which was shown in Figures 1 and 2 above), appeared to capture a key set of interfaces between the system and the organisation in which it is embedded – those interfaces which, when changed, lead to changes in the scope, and vide versa. The interfaces do not uniquely and completely define the system, but they do appear to provide a source of crucial information for identifying and resolving the discordances that were discussed in the background section above, and which are the focus of this research.

Merging the above two threads of previous research provides a clear approach to supporting the automated identification of those discordances between an IT system and its environment that have the potential to affect decisions on the system scope and its corresponding boundary. These identified discordances can then be used to raise issues within a linked issue-tracking system. The issues, when resolved, will allow a progressive clarification and elaboration of the agreed system scope (as well as the concomitant changes to the associated organisational workflows and business processes).

The approach will be based on the development of a modelling notation which links information flows, functional boundaries and work flows – and which supports the development of models that initially represent the current (*as-is*) domain. The models are then progressively adjusted to show the incorporation of variations on the proposed IT system, with discordances being automatically identified and issues being raised with the developer as the models are modified. The value of a visual notation that can be evolved can be seen in the success of approaches such as Threat Model Analysis (TMA). Visualising the system helps to make the boundaries more readily apparent, and thus less likely to be overlooked. The existence of the model also allows the impact of changes in the relationships between elements on the security threats to the system to be assessed and addressed. A prototype tool which forms part of this research will allow evaluation of the effectiveness of the models and algorithms in managing the construction and evolution of a system, and identifying boundary conditions that require negotiation between the client and the developer. We believe this approach will lead to a much more rapid convergence of the agreed system scope and a substantial reduction in overlooked adverse misalignment between the system and its environment.

Approach

As was mentioned above, a key characteristic of much software systems development is early uncertainty in the system scope. There exists a significant and growing body of research into the early stages of the development cycle when the development scope is nominally resolved – mostly focussing on domain modelling and requirements capture and analysis. Most of this research however assumes an initially unknown but largely invariant system scope which simply must be elicited and analysed. Both research and practice often overlooks the complex interdependencies through which the emerging definition of a system can directly or indirectly affect the environment in which that system exists, and thereby create a feedback loop which leads to subsequent changes in the scope of that system [21]. This type of interdependency between system and environment is most common where part of a business process is being fundamentally changed or replaced by a software

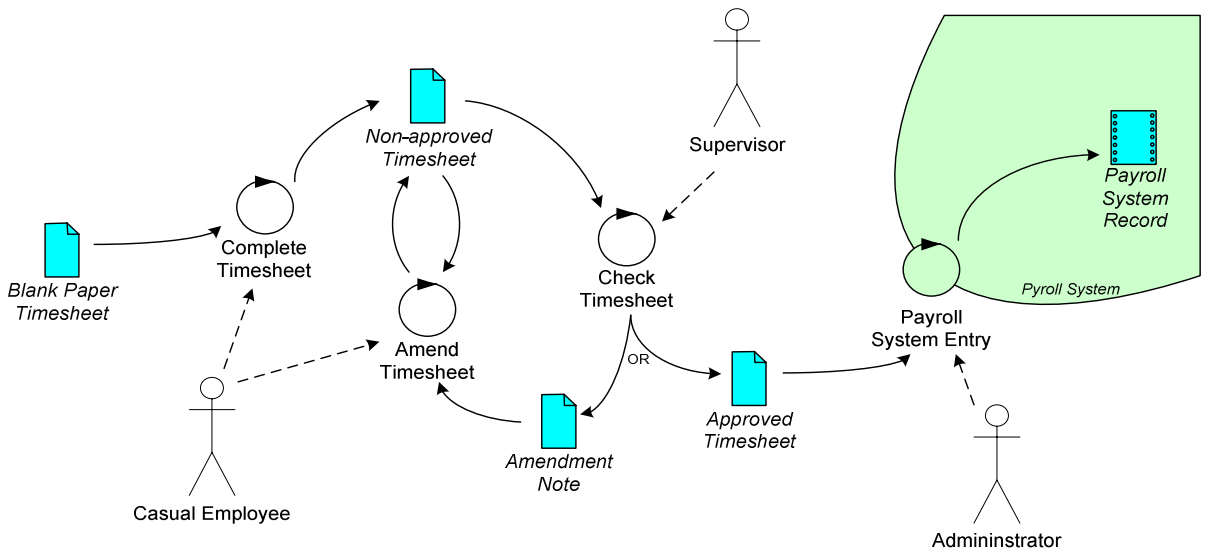

Figure 1: Existing casual staff payment process

system. Whilst this characteristic of system development is not uncommon in most (if not all) software systems, recent technologies, and especially web-based systems, are exemplars of where the scale of the feedback mechanism makes addressing it early an imperative.

Before we consider existing research which is relevant, let us illustrate the above problem with a simplified illustrative example. Consider an existing business process (variants of which exist in many small businesses) that involves casual employees completing a paper-based timesheet at the end of each week which is subsequently checked by a supervisor and either returned for amendment or approved and submitted to a payroll administrator for payment. This is shown in Figure 1 in a simplified process modelling notation, in which the shaded region represents an existing payroll IT system. This figures adopts a simplistic notation, but is used to illustrate the representation of the relationship between the business workflows and the existing (or anticipated) software systems. The modelling notation to be developed will be much more sophisticated than shown in this figure.

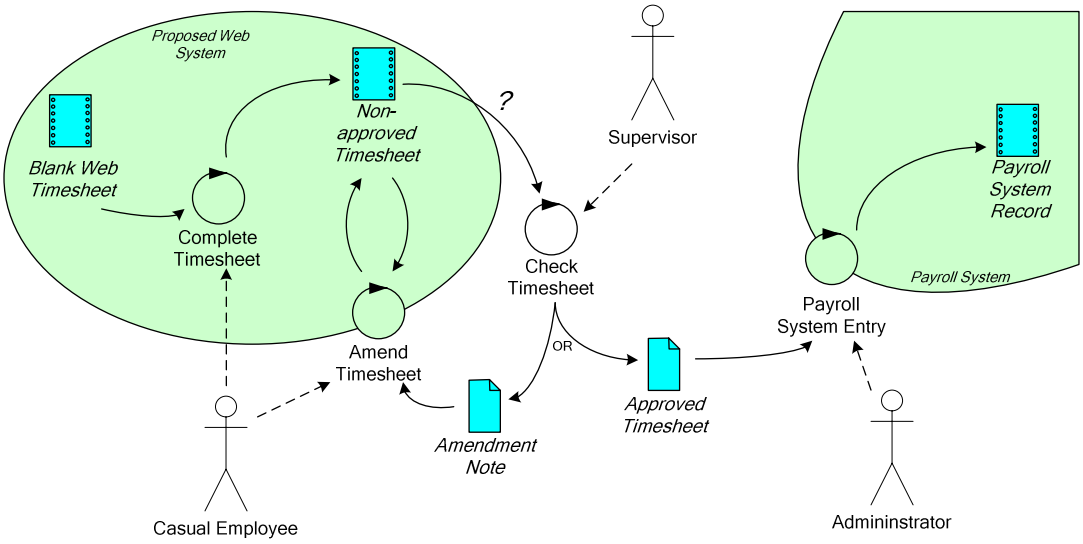

Figure 2: **Initial scope of electronic timesheet submission**

Assume that a decision was then taken to develop a simple system to support online submission (and modification, when necessary) of timesheets by casual employees. The initial system scope was focussed on just support for the casual employees, and could be defined as shown in Figure 2. Such a change in the relevant subsection of the workflow would however have impacts on – and potentially require changes to – other sections of the overall business process. For example, the processing of timesheets by the staff supervisor was previously carried out on the paper-based timesheets, but these no longer exist in that form (now being

electronic). The result is a discordance between the new software system and part of the existing process (the checking of the timesheets which assumes paper-based input). Resolving this discordance will involve changes to the scope of the proposed system and/or the business processes, and would be the basis for negotiation with the client.

In other words, when we move from a system as it currently exists (the as-is view of the business/system) to the definition of a potential new or changed system (the to-be view of the business/system) we introduce potential discordances between the system and the business processes to be supported by the new system. Resolving these discordances leads either to changes in the scope of the system, or changes in the associated business processes – either of which can lead to further changes to either or both. Much of the complexity of the early stages of project scoping resolves around understanding and negotiating these changes and defining the boundary of what functionality should "the system to be" support and which it should not. Whilst this form of scope resolution is well accepted and understood, it is our contention that it is invariably overlooked, and there has been little research specifically addressing how it can be most effectively managed. We contend that by undertaking richer forms of the modelling shown in the above examples, based on a merger of existing process modelling techniques and our own information flow modelling formalism [16], it is possible to automatically identify potential discordances early in the project scoping / specification, and to then raise these as development issues that need to be resolved – thereby leading to more rapid convergence onto an agreed system scope which is integrated with the modified organisational workflows and business processes.

As mentioned above, there has been substantial research over the last 30+ years, focusing on the early stages of software systems development and the relationships between software systems and business processes. Requirements Engineering (RE) is an active research area which focuses on approaches to capturing, analysing and modelling requirements for software systems. The majority of research in this field assumes a fixed, though initially unknown, scope which must be discovered, analysed and documented. Where requirements are recognised as varying (or volatile [22]) this is usually attributed to uncertainty on the part of the client [13] or changes occurring in the domain independently of the introduction of the *system-to-be*. Very little research has considered the way in which the introduced system itself can lead to changes in the domain – and hence create a positive feedback loop leading to changes in the system. Our earlier research, which has had a specific focus on Web systems development, has shown that often Web systems and the organisation domain in which the systems exists co-evolve, with the nature and extent of this evolution often not being clearly understood until early design prototypes are available [8,9].

In many respects, this research is also closely related to work on IT-business alignment, which focuses on ways in which a software system can be most appropriately designed to seamlessly integrate with, and support, existing or proposed business activities. Of particular interest is research on *strategic alignment* [15]. Research has shown that strategic alignment can have substantial positive impact on business performance [4]. Whilst the desired end result is similar (the absence of discordances between the system and the business processes – or the business objectives which are supported by those processes), the focus of work on IT-business alignment is typically on how to ensure that software systems appropriately support a given set of business objectives, rather than the identification of specific aspects where an existing business process requires modification as a consequence of the introduction or modification of a software system to which it interfaces. Furthermore, the research in this area has not paid due attention to the nature and extent of impact that the "*system-to-be*" may have on the corresponding business rules that govern business processes and workflows within organisations.

Similarly, work in areas as diverse as soft-systems methodologies (SSM) [3], problem frames [7] and COTS (Commercial Off-The-Shelf) development [18] also provides insights into the interdependence of software systems and the organisational processes within which they are embedded. For example, rich pictures – a tool used within SSM and elsewhere – can be used to understand the relationships between software systems and the contexts within which they exist. Nevertheless, again, these techniques are useful in supporting effective system design, but not in identifying specific points of discordance. This identification is typically assumed to occur as a natural consequence of the system design process, and hence has lacked any focus as a research topic. Indeed, this assumption is partially true – the discordances do indeed become obvious – but often not until later in the development cycle, well after design or indeed implementation has commenced and scoping contracts have been agreed upon and signed off.

Given that there is much work on what defines a software system effectively aligned with organisational processes or business goals, but not on specific techniques for identifying particular points where they are potentially misaligned, the obvious question is how such discordances can be discovered as early in the development as possible and what strategies could be employed in early resolution of issues arising from the identification of these discordances. Answering these questions is the core of this research project.

Our particular approach to answering this question emerges from the convergence of two of our earlier research contributions: investigation of the ways in which issues that emerge during software system design (and Web development in particular) have influenced the developers understanding of the system scope [10,21]; and the potential role in web system design of a high-level information flow model [16]. Taken together, these two areas of work indicate the importance to defining system scope of understanding the flow of information into and out of a system, especially when coupled with an equivalent process flow. These areas of work are the main motivations behind the development of the AWeSOMe tool.

Methodology

The AWeSOMe tool is being developed to support investigation of issues around the management of scope and requirements of web systems. The design decisions made when developing AWeSOMe have been guided by these research goals. A brief discussion of these goals follows to show how they have influenced the design of AWeSOMe.

Stage 1: Analysis of existing data on issue resolution and scope refinement: In the first stage of the project, we will consider the question of what system/domain interfaces are associated with those discordances that, when resolved, lead to scope changes. We will analyse the data collected in our earlier research on issue analysis with the specific objective of identifying those issues which related to an identified discordance. These issues will then be analysed to select only those which resulted in a subsequent change or refinement to the perceived or agreed system scope. The AWeSOMe tool provides a realisation of the concepts being discussed in this phase and provides a focal point for discussions. Work on AWeSOMe feeds back into these discussions.

Stage 2: Development of a modelling notation for representing interfaces between a software system and its business domain: Using results from our previous work [16,17] we will develop a rich model that captures the way in which business models and processes inter-relate with IT system designs, particularly in terms of their mutual impacts. This model will be compatible with existing business and system design models (UML for example) and will leverage work that focuses on the impacts IT can have on organisational operation. The modelling notation will be lightweight yet expressive, and be understandable to both developers and clients – thereby facilitating communication between them. Of specific interest will be the i* modeling notation and the corresponding framework developed by Yu [19]. The purpose of this modelling is to support identification of misalignments that exist between the IT systems' core functionality and the organisational workflows (see stage 3). AWeSOMe acts as tool for prototyping different notations and experimenting with different approaches to modelling. Because it is not tied to any existing notations or frameworks we are free to try out new concepts.

Stage 3: Automated discordance identification algorithms: In this stage we will develop algorithms for analysing the interface models constructed using the notation developed in stage 2. These algorithms will be based on identifying points in the models where the specific modelling semantics have been breached by making changes to the models. In particular, as a model is constructed or changed to incorporate the relationship between a proposed software system and the organisational workflows it supports, the algorithms should be able to identify discordances in these interfaces that have the potential, when resolved, to affect the system scope. We will investigate existing automated reasoning algorithms and technologies to find the most effective one for our purposes. The models created using the AWeSOMe tool are stored in a database in a simple schema that we have created. Using this data model we can create software components that analyse this data using the algorithms developed in this phase. These tools can be standalone or they can be integrated with AWeSOMe.

Stage 4: Implementation of tools to support reasoning: In this component we will develop a prototype tool which supports the construction and evolution of the above models that is suitable for use in industry-based case studies. The AWeSOMe tool will serve as the basis for this implementation, both conceptually and technically. The prototype tool will be interoperable with existing CASE tools so that the models can also form the basis of subsequent modelling (e.g. the models should be able to be exported into skeletons of preliminary UML use case diagrams). Our preliminary design for this tool involves an interface that supports the construction of the *as-is* model (i.e. existing workflows and information flows) and the subsequent modification of the model to incorporate proposed changes and new system functionality (the *to-be* model). A rich versioning capability will allow the modeller to wind back or forward the model (or different versions of the changes) in order to consider possible ways in which the scope can be affected. We will also implement in the tool the discordance identification and subsequent issue handling processes. The result will be a composite modelling, versioning and issue tracking CASE tool prototype.

Stage 5: Evaluation and refinement: Supporting all the above project components, and running throughout the project, is a series of user and usage experiments. We will conduct experiments with commercial developers using the model and associated tool to track the extent to which they support effective identification of discordances and raises these as issues to be resolved. Specifically we will consider the extent to which the identified discordances are valid and complete, and the extent to which their resolution affects the agreed system scope (or at least led to valid discussions about the scope). We will also conduct case studies of the use of the tool. Evaluations will be performed both with AWeSOMe and with the tool developed in stage 4.

Conceptual Framework

A key purpose of the AWeSOMe tool is to assist with the development of new techniques and methods for building web systems. These techniques and methods are based on the existing body of work relating to Web Engineering. In particular we build on the concept of alignment in systems modelling [1,2,4], existing approaches to modelling systems [14], and existing notations used for modelling systems [14]. The use of these concepts in this work is introduced below.

Alignment & Discontinuities

A system can be aligned, misaligned, or unaligned. An unaligned system can not be realised. It is a system that has discontinuities. A misaligned system is one where there are gaps between the web system and the organisational processes it is supporting, but these gaps have been identified and steps have been taken to ensure the process still works. Sometimes a physical change such as printing hard copy versions of electronic documents is an acceptable workaround. When a discontinuity is resolved with a workaround rather than with a change to the software scope we say there is a misalignment, but the discontinuity has been resolved. An aligned system is one where the web system totally encompasses the organisational processes. A discontinuity is a discordance that has not been resolved. A model is aligned when all its discontinuities have been resolved. A discordance can be changed either by modifying the software system or the physical system. AWeSOMe is developed to help developers of web systems identify these discordances early in the development cycle, when the cost of rectifying them is lower than it would be otherwise.

Notations

A key component of this research is developing a notation that can be used to create models of web systems and organisations. There are numerous notations available for modelling web systems and our own research has also developed a useful notation [16]. The AWeSOMe tool seeks to build on this research and support further research in to notations that are useful for modelling web systems. In particular, we believe it is essential for many projects to have a simple notation that can readily be applied to real world problems. While BPMN, BPEL, IDEF(n) and the like are extremely useful, there is often a great deal of overhead and cost involved in integrating these modelling notations and techniques into an organisation. We aim to develop a simple notation that will be easily adopted for projects of any size, while still retaining the richness of expression that makes other notations useful.

An example of the kind of notation we are developing is shown in Figures 1&2. This notation describes a system in terms of Actors, Activities, and Artefacts, and the flows between them. The flows indicate, for example, that an Actor can perform an activity, and that an Artefact can be the input or output of an Activity. This notation is not well developed, but with the AWeSOMe tool serving as a prototype we can easily modify notations and evaluate how useful they are for modelling actual systems.

Software Framework

The concepts developed in this work are intended to be applied to real world problems. While many of the concepts are widely used in existing tools, we believe our approach to integrating them to be unique. For this reason we have chosen to develop a software framework that supports the creation of tools based on the ideas developed in this work. These tools aim to be useful both as research aids as we develop our ideas, and as tools that can be used in web engineering projects. The following discussion presents our approach to AWeSOMe's architecture and a brief discussion of its key components.

4-Layer Architecture

In order to support our research goals we have chosen to base AWeSOMe on a 4-layer architecture as used in UML. Figure 4 shows a linear representation of the UML 4-layer architecture on the left and the corresponding layers used in the AWeSOMe architecture.

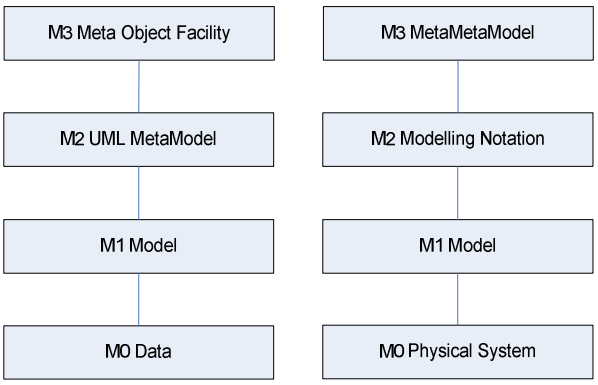

Figure 3 4-Layer Architectures for UML and AWeSOMe.

The UML layers are described in the UML 2.0 specification [cite]. In the AWeSOMe architecture, the significance of the layers is as follows:

- M0 – the physical system that is being modelled. This can be an implementation of a web system, or the processes and workflows of a business.
- M1 – a model representing the physical system. The model is an abstraction that allows aspects of the physical system to be represented in a data structure that supports reasoning about the system. The model is expressed using a notation.
- M2 – a notation for describing models of organisations or web systems. Common examples of the kinds of notations describe are BPMN, UML Activity Diagrams, and WIED.
- M3 – a model for describing notations. The M3 layer is constructed to be as simple as possible while still allowing a variety of notations to be expressed. The M3 layer describes notations in terms of the entities that can be used in the models and the relationships that can be created between these entities. The M3 layer is expressed as a database schema.

Modelling Framework

The modelling framework is a collection of database schemas, software interfaces and component designs that support the implementation of a notation independent tool for modelling web systems and organisations. Figure 5 shows how the 4-layer architecture is realised as

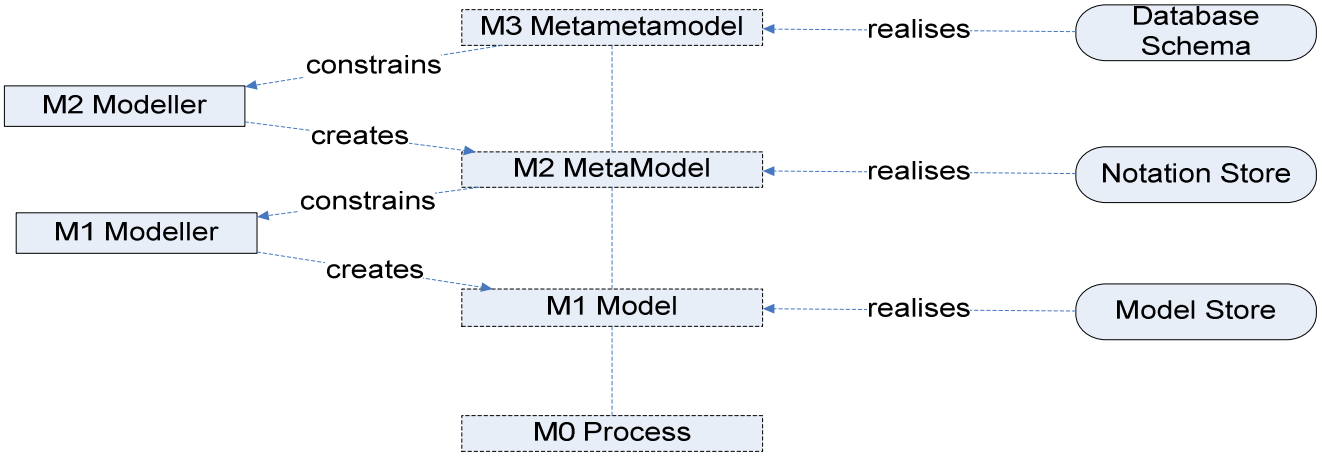

Figure 4 The AWeSOMe modelling framework

Database Schema

The M3 database schema describes a way of representing notations. We have kept this description as simple and as minimally abstract as possible. In our schema we say that a Notation consists of Entities and Relationships. Both of these can have Attributes associated with them. We also support the notion of superclasses and abstract Entities. In the example notation described above, the Entities are activities, actors, and artefacts. The Relationships are invoke and produce. Relationships are defined as having to and from entities, so when modelling the notation we would say the from-Entity for invoke is actor and the to-Entity for invoke is artefact.

M2Modeller

The M2Modeller allows us to model notations that are used to model web systems or organisations (or both). Modelling different notations allows us to experiment with different approaches to modelling web systems and organisations. Our goal is to develop a notation that is simple to use but effective for modelling web systems and organisations. The M2Modeller allows us to capture modelling semantics and layout semantics. In our implementation the M2Modeller is a web application developed in Java using the Struts framework and Hibernate. Notations created in the M2Modeller are stored in the Notation Store. The Notation Store is implemented as a Hibernate persistence layer, and in the current implementation also uses a MySQL database.M1Modeller

M1Modeller

The M1Modeller allows us to model web systems and organisations using any of the notations modelled in the M2Modeller. It also allows us to manage branches and evolutions in the model. The M1Modeller will also be used to interact with software components that help to resolve alignment discontinuities in the model being created. In our implementation the M1Modeller is a 2-tier client-server application. The server layer consists of a collection of web-services that manage the M1 data and layout models. This layer is implemented using Apaches Axis web-service framework and Hibernate. These services are consumed by a C# application that allows the user to create and manipulate models of physical systems using notations that have been added to the system. The model created using the M1Modeller is stored in the Model Store. The Model Store is implemented as a Hibernate persistence layer, and in the current implementation also uses a MySQL database. Figure 5 shows an example of modelling the timesheet system discussed earlier using AWeSOMe's M1Modeller.

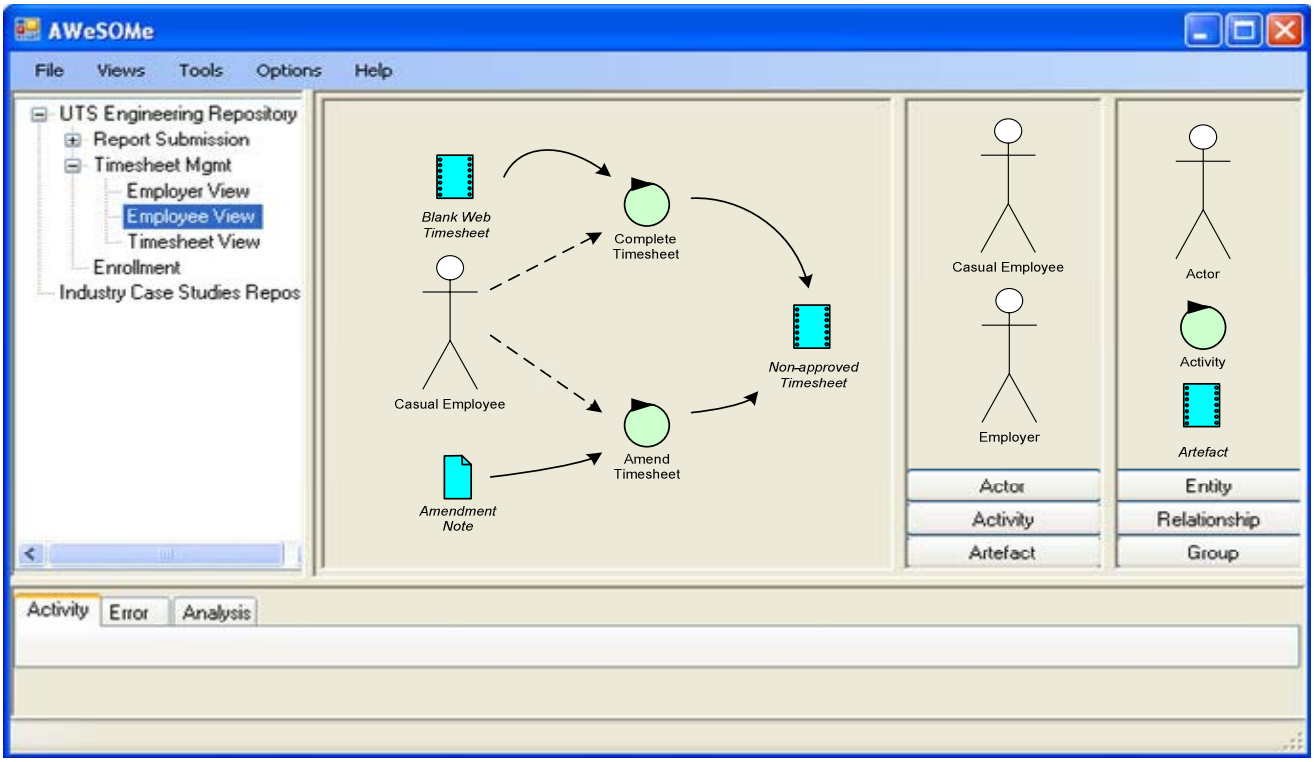

Figure 5 The M1Modeller for AWeSOMe

Further Work

The AWeSOMe modeller is being developed to support ongoing research into the development of Web Systems. This research intends to have several outcomes, which will be actualised in the AWeSOMe tool and in related tools the investigation identifies as being necessary. These outcomes are:

- An improved understanding (represented as models) of the interdependence of organisational workflows and the software systems which support these workflows, and particularly with the way in which appropriate modelling (based on a composite of workflow modelling and information flow modelling) of these interdependencies can lead to identification of discordances between the workflows and the systems.
- A set of techniques for undertaking reasoning about the models that are constructed and the subsequent automated identification of those discordances between software systems and the workflows that are supported which have the potential to affect, when resolved, the agreed scope of the software system.
- A prototype tool which supports the construction of the models and subsequent reasoning about these models, and which thereby assists in identifying misalignments between software systems and the workflows that they support.
- Integration of the prototype tool with existing widely-utilised software product development tools which thereby allow the rapid adoption and leveraging of the models and techniques which will be developed.

References

1. Aversano, L., Bodhuin, T., and Tortorella, M. (2005), “Assessment and impact analysis for aligning business processes and software systems”, In *Proceedings of the 2005 ACM Symposium on Applied Computing* (Santa Fe, New Mexico, March 13 - 17, 2005). ACM Press, New York
2. Bleistein S.J.,, Aurumn A., Cox K., and Ray P., “Strategy-Oriented Alignment in requirements engineering: linking business strategy to requirements of e-business systems using the SOARE approach”, *Journal of Research and Practice in Information Technology*, 36(4): 259-276, 2004.
3. Checkland, P., & Scholes, J. (1990). *Soft systems methodology in action*. Toronto: Wiley.
4. Croteau A.M., “Organisational and technological infrastructures alignment”, Proc *of the 34th Hawaii International Conf. on Systems Sciences*, IEEE, 2001.
5. X Fu, T Bultan, J Su (2004), “Analysis of interacting BPEL web services”, *Proceedings of the 13th conference on World Wide Web*, May 17-22, 2004, New York

6. Houghton, J.W. (2005), “Australian ICT Trade Update-2005”, Centre for Strategic Economic Studies, ACS, Australia. Available Online at: https://www.acs.org.au/members/index.cfm?action=notice&notice_id=376
7. Jackson, M (2000) *Problem frames: analyzing and structuring software development problems.* Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA
8. Kong, X., Liu, L., & Lowe, D. (2005, 5-7 Dec). “Modeling an agile web maintenance process using system dynamics*”. 11th ANZSYS / Managing the Complex V conference*, Christchurch, NZ.
9. Kong, X., Liu, L., & Lowe, D. (2005, 18-21 Oct). *“Supporting Web User Interface Prototyping through Information Modeling and System Architecting”. ICEB'2005: IEEE Intern. Conf. on e-Business Engineering*, Beijing, China.
10. Lowe D and Webby R. (1998) ‘Web Development Process Modelling and Project Scoping’ in *Proc of First International Workshop on Web Engineering (WebEng 98)*, 14 April 1998, Brisbane, Australia
11. Lowe D, (2000) ‘A Framework for Defining Acceptance Criteria for Web Development Projects’, *Second ICSE Workshop on Web Engineering*, 4 and 5 June 2000, Limerick, Ireland
12. Lowe D., Yusop N., Zowghi D. (2003) ‘Using Prototypes to Understand Business Impacts of Web Systems;, *Proceedings of the 8th Australian World Wide Web conference*, Gold Coast, July 2003.
13. Lowe, D., & Eklund, J. (2003). Client Needs and the Design Process in Web Projects. *Journal of Web Engineering, 1*(1), 23-36.
14. Lowe, D & Tongrungrojana, R (2004), 'Web Information Exchange Diagrams for UML', *WISE 2004: The 5th Intern.l Conf. on Web Information Systems Eng.*, vol. LNCS 3306, Springer, Brisbane, Aust., pp. 29-40.
15. McKeen, J. D., and Smith H., “Making IT happen: Critical issues in IT management”, Chichester; Hoboken, NJ, Wiley 2003.
16. Tongrungrojana, R., & Lowe, D. (2004). WebML+: A Web Modeling Language for Modelling Architectural-Level Information Flows. *Journal of Digital Information, 5*(2), (online journal).
17. Tongrungrojana, R., & Lowe, D. (2004, 3-7 July). *Understanding business impacts of changes to information designs: A case study.* AusWeb04: The 10th Austr. World Wide Web Conference, Gold Coast, Australia.
18. Torchiano, M. and Morisio, M., “Overlooked aspects of COTS development”, IEEE Software March/April 2004: 88-93.
19. Yu., E., (1997), “Towards modelling and reasoning support for early-phase requirements engineering”, Proc of the IEEE Int. Symposium on Requirements Engineering, RE97.
20. Yusop N., Zowghi D., and Lowe D. (2003), ‘An Analysis of E-Business Systems Impacts on the Business Domain’, *Procs of the 3rd Intern. Conf. on Electronic Business (ICEB 2003)*, Singapore, December 2003.
21. Yusop, N, Lowe, D & Zowghi, D (2005), 'Impacts of Web Systems on their Domain', *Journal of Web Engineering*, vol. 4, no. 4, pp. 313-38.
22. Zowghi D., Offen, R. Nurmuliani, (2000) ‘The Impact of Requirements Volatility on the Software Development Lifecycle’, *Proc. of the Int. Conf. on Software, Theory and Practice (ICS2000)*, China.